Weighted Model Counting with Conditional Measures

Paulius Dilkas

21st July 2020

1 Introduction

- The Main Narrative
  1. When weights are defined on literals, the measure on the free BA is fully independent.
  2. This means that the BA itself must be larger (i.e., have additional 'meaningless' literals) to turn any probability distribution into an independent one.
  3. We show how we can define conditional weights on literals, allowing us to encode any probability distribution into a Boolean algebra that's not necessarily independent and thus can be smaller.
  4. We demonstrate a specific example of this by presenting a new way to encode Bayesian networks into instances of WMC and adapting a WMC algorithm (ADDMC) to run on the new format.
  5. We show that this results in significantly faster inference.
  6. We show that our encoding results in asymptotically fewer literals and fewer ADDs, and thus a simpler problem.
  7. (Maybe) we experimentally demonstrate a phase transition based on the number of variables per ADD.
- Potential criticism may be that this is a step backwards and doesn't allow us to use SAT-based techniques for probabilistic inference. However, they can still be used for the 'theory+query' part.
  - Zero-probability weights and one-probability weights can be interpreted as logical clauses. This doesn't affect ADDMC but could be useful for other solvers.

F What are the main claims, what are the main takeaways, intuitive [???] of theorems to follow. To do this, we appeal to algebraic constructions to define the main concepts for introducing measures on Boolean algebras.

- Algorithms[1]
  - ADDMC [18] (rediscovered the multiplicativity of BAs in different words) (with optimal settings)
  - Cachet [44]
  - c2d [16]
  - d4 [36] (closed source, boo!)
  - miniC2D [41]
- Notable previous/related work
  - Hailperin's approach to probability logic [25]

[1] `http://beyondnp.org/pages/solvers/model-counters-exact/`

Table 1: (in the same order)

	Set-theoretic notation	Boolean-algebraic notation
Atoms (elements of U)	a, b	a, b
Models (elements of 2^U)	$\emptyset, \{a\}, \{b\}, \{a, b\}$	$\neg a \wedge \neg b, a \wedge \neg b, \neg a \wedge b, a \wedge b$
	$\{\emptyset, \{a\}, \{b\}, \{a, b\}\}$	$\top$
	$\{\emptyset, \{a\}, \{b\}\}, \{\emptyset, \{a\}, \{a, b\}\}$	$\neg a \vee \neg b, a \to b$
	$\{\emptyset, \{b\}, \{a, b\}\}, \{\{a\}, \{b\}, \{a, b\}\}$	$b \to a, a \vee b$
Formulas (elements of 2^{2^U})	$\{\emptyset, \{a\}\}, \{\emptyset, \{b\}\}, \{\emptyset, \{a, b\}\}$	$\neg b, \neg a, a \leftrightarrow b$
	$\{\{a\}, \{b\}\}, \{\{a\}, \{a, b\}\}, \{\{b\}, \{a, b\}\}$	$(a \wedge \neg b) \vee (b \wedge \neg a), a, b$
	$\{\emptyset\}, \{\{a\}\}, \{\{b\}\}, \{\{a, b\}\}$	$\neg a \wedge \neg b, a \wedge \neg b, \neg a \wedge b, a \wedge b$
	$\emptyset$	$\bot$

  - Nilsson's (somewhat successful) probabilistic logic [39, 40]
  - Logical induction: a big paper with a good overview of previous attempts to assign probabilities to logical sentences in a sensible way [22]
  - Measures on Boolean algebras
    * On possibility and probability measures in finite Boolean algebras [7]
    * Representation of conditional probability measures [34]

- Intuitively, a measure is just like a probability, except it's in $\mathbb{R}_{\geq 0}$ instead of $[0, 1]$.
- ADDMC should probably be in textsc mode.

2 Preliminaries

- Make up my mind about a, b vs. x, y and stick to it (maybe x, y?).
- Terminology: 'with generating set S' $\to$ 'over S'.
- Notation: if L denotes literals, then it doesn't denote a generating set. Literals should be U.

Definition 1. A *Boolean algebra* (BA) is a tuple $(\mathbf{B}, \wedge, \vee, \neg, 0, 1)$ consisting of a set $\mathbf{B}$ with binary operations *meet* $\wedge$ and *join* $\vee$, unary operation $\neg$ and elements $0, 1 \in \mathbf{B}$ such that the following axioms hold for all $a, b, \in \mathbf{B}$:

- both $\wedge$ and $\vee$ are associative and commutative;
- $a \vee (a \wedge b) = a$, and $a \wedge (a \vee b) = a$;
- 0 is the identity of $\vee$, and 1 is the identity of $\wedge$;
- $\vee$ distributes over $\wedge$ and vice versa;
- $a \vee \neg a = 1$, and $a \wedge \neg a = 0$.

For clarity and succinctness, we will occasionally use three other operations that can be defined using the original three[2]:

$$\begin{aligned} a \to b &= \neg a \vee b, \\ a \leftrightarrow b &= (a \wedge b) \vee (\neg a \wedge \neg b), \\ a + b &= (a \wedge \neg b) \vee (\neg a \wedge b). \end{aligned}$$

[2]We use + to denote symmetric difference because it is the additive operation of a Boolean ring.

We can also define a partial order $\leq$ on $\mathbf{B}$ as $a \leq b$ if $a = b \wedge a$ (or, equivalently, $a \vee b = b$) for all $a, b \in \mathbf{B}$. Furthermore, let $a < b$ denote $a \leq b$ and $a \neq b$. For the rest of this paper, let $\mathbf{B}$ refer to the BA $(\mathbf{B}, \wedge, \vee, \neg, 0, 1)$. For any $S \subseteq \mathbf{B}$, we write $\bigvee S$ for $\bigvee_{x \in S} x$ and call it the *supremum* of S. Similarly, $\bigwedge S = \bigwedge_{x \in S} x$ is the *infimum*. By convention, $\bigwedge \emptyset = 1$ and $\bigvee \emptyset = 0$. For any $a, b \in \mathbf{B}$, we say that a and b are *disjoint* if $a \wedge b = 0$.

Definition 2 ([29, 37])**.** An element $a \neq 0$ of $\mathbf{B}$ is an *atom* if, for all $x \in \mathbf{B}$, either $x \wedge a = a$ or $x \wedge a = 0$. Equivalently, $a \neq 0$ is an atom if there is no $x \in \mathbf{B}$ such that $0 < x < a$. We say that $\mathbf{B}$ is *atomic* if for every $a \in \mathbf{B} \setminus \{0\}$, there is an atom x such that $x \leq a$.

Lemma 1 ([21])**.** *For any two distinct atoms $a, b \in \mathbf{B}$, $a \wedge b = 0$.*

Lemma 2 ([23])**.** *The following are equivalent:*

- $\mathbf{B}$ *is atomic.*
- *For any $x \in \mathbf{B}$, $x = \bigvee_{atoms\ a \leq x} a$.*
- 1 *is the supremum of all atoms.*

Lemma 3 ([23])**.** *All finite BAs are atomic.*

Definition 3 ([20, 29])**.** A *measure* on $\mathbf{B}$ is a function $m \colon \mathbf{B} \to \mathbb{R}_{\geq 0}$ such that:

- $m(0) = 0$;
- $m(a \vee b) = m(a) + m(b)$ for all $a, b \in \mathbf{B}$ whenever $a \wedge b = 0$.

If $m(1) = 1$, we call m a *probability measure*. Also, if $m(x) > 0$ for all $x \neq 0$, then m is *strictly positive*.

Lemma 4 ([47])**.** *For any $a, b \in \mathbf{B}$, $a \leq b$ if and only if $a \wedge \neg b = 0$.*

Lemma 5 ([23])**.** *Let $m \colon \mathbf{B} \to \mathbb{R}_{\geq 0}$ be a measure. Then for all $a, b \in \mathbf{B}$, if $a \leq b$, then $m(a) \leq m(b)$.*

2.1 The Space of Functions on Boolean Algebras

Definition 4 (Operations on functions)**.** Let $A \colon 2^X \to \mathbb{R}_{\geq 0}$ and $B \colon 2^Y \to \mathbb{R}_{\geq 0}$ be Boolean functions, $\alpha \in \mathbb{R}_{\geq 0}$, and $x \in X$. We define the following operations:

Addition: $A + B$ is a function $A + B \colon 2^{X \cup Y} \to \mathbb{R}_{\geq 0}$ such that

$$(A + B)(\tau) = A(\tau \cap X) + B(\tau \cap Y)$$

for all $\tau \in 2^{X \cup Y}$.

Inverse: $\overline{A}$ is a function $\overline{A} \colon 2^X \to \mathbb{R}_{\geq 0}$ such that

$$\overline{A}(\tau) = 1 - A(\tau)$$

for all $\tau \in 2^X$.

Multiplication: $A \cdot B$ is a function $A \cdot B \colon 2^{X \cup Y} \to \mathbb{R}_{\geq 0}$ such that

$$(A \cdot B)(\tau) = A(\tau \cap X) \cdot B(\tau \cap Y)$$

for all $\tau \in 2^{X \cup Y}$.

Scalar multiplication: αA is a function $\alpha A \colon 2^X \to \mathbb{R}_{\geq 0}$ such that

$$(\alpha A)(\tau) = \alpha \cdot A(\tau)$$

for all $\tau \in 2^X$.

Projection: $\exists_x A$ is a function $\exists_x A\colon 2^{X\setminus\{x\}} \to \mathbb{R}_{\geq 0}$ such that

$$(\exists_x A)(\tau) = A(\tau) + A(\tau \cup \{x\})$$

for all $\tau \in 2^{X\setminus\{x\}}$.

Observation 1. *Let $\mathcal{V} = \{A\colon 2^X \to \mathbb{R}_{\geq 0} \mid X \subseteq U\}$. Then $\mathcal{V}$ is a semi-vector space with three additional operations: inverse, (non-scalar) multiplication, and projection. Specifically, note that both addition and multiplication are both associative and commutative.*

Definition 5 (Special functions)**.**

- unit $1\colon 2^{\emptyset} \to \mathbb{R}_{\geq 0}$, $1(\tau) = 1$.
- zero $0\colon 2^{\emptyset} \to \mathbb{R}_{\geq 0}$, $0(\tau) = 0$.
- constant $[a]\colon 2^{\{a\}} \to \mathbb{R}_{\geq 0}$,

$$[a](\tau) = \begin{cases} 1 & \text{if } a \in \tau \\ 0 & \text{if } a \not\in \tau. \end{cases}$$

Remark. For any function $A\colon 2^X \to \mathbb{R}_{\geq 0}$, $A + \overline{A} = 1$.

Henceforth, for any function $A\colon 2^X \to \mathbb{R}_{\geq 0}$ and any set τ, we will write $A(\tau)$ to mean $A(\tau \cap X)$.

3 Weighted Model Counting as a Measure

F2 Explain with examples: models = elements [atoms] of algebra.

- Be careful about mentioning ideals and quotients.
- Note that we're going to use logical notation for BAs. Give an example of the two notations.

Example 1. Let $\mathcal{L}$ be a propositional logic with p and q as its only atoms. Then $L = \{p, q, \neg p, \neg q\}$ is its set of literals. Let $w : L \to \mathbb{R}_{\geq 0}$ be the *weight function* defined by

$$\begin{aligned} w(p) &= 0.3, \\ w(\neg p) &= 0.7, \\ w(q) &= 0.2, \\ w(\neg q) &= 0.8. \end{aligned}$$

Let Δ be a theory in $\mathcal{L}$ with a sole axiom p. Then Δ has two models, i.e., $\{p, q\}$ and $\{p, \neg q\}$. The *weighted model count* (WMC) [12] of Δ is then

$$\sum_{\omega \models \Delta} \prod_{\omega \models l} w(l) = w(p)w(q) + w(p)w(\neg q) = 0.3.$$

The corresponding BA $B(\Delta)$ can then be constructed as a Lindenbaum-Tarski algebra. Alternatively, one can first construct the free BA generated by the set $\{p, q\}$ and then take a quotient with respect to either the filter generated by p or the ideal[3] generated by $\neg p$.

Each element of $B(\mathcal{L})$ can also be seen as a subset of the set of all models of $\mathcal{L}$, with 0 representing $\emptyset$, 1 representing the set of all (four) models, each atom representing a single model, and each edge going upward representing a subset relation. Thus, the Boolean-algebraic way of calculating the WMC of Δ consists of:

[3]More details on these concepts can be found in many books on BAs [23, 33].

1. Identifying an element $a \in B(\mathcal{L})$ that corresponds to Δ.
2. Finding all atoms of $B(\mathcal{L})$ that are 'dominated' by a according to the partial order.
3. Using w to calculate the weight of each such atom.
4. Adding the weights of these atoms.

This motivates the following definition of WMC generalised to BAs.

- Why is Step 1 always possible?
- Clarify what $B(L)$ means and whether $B(\Delta)$ is even necessary.
- Find a reference for the set/subset thing.

Definition 6. Let U be an arbitrary set. Then

- a *measure* is a function $M\colon 2^{2^U} \to \mathbb{R}_{\geq 0}$,
- a *weight function* is a function $W\colon 2^U \to \mathbb{R}_{\geq 0}$.
- a weight function is *factored* if $W = \prod_{x \in U} W_x$ for some functions $W_x\colon 2^{\{x\}} \to \mathbb{R}_{\geq 0}$.
- every weight function induces a measure:

$$M_W(x) = \begin{cases} 0 & \text{if } x = \emptyset \\ W(u) & \text{if } x = \{u\} \\ \sum_{\{u\} \leq x} W(u) & \text{otherwise.} \end{cases}$$

  The process of calculating the value of $M_W(x)$ for some $x \in 2^{2^U}$ with a given definition of W is known as *weighted model counting*.

Given a theory Δ in a logic $\mathcal{L}$, the usual way of using WMC to compute the probability of a query q is [5, 45]

$$\Pr_{\Delta,w}(q) = \frac{\mathrm{WMC}_w(\Delta \wedge q)}{\mathrm{WMC}_w(\Delta)}.$$

In our algebraic formulation, this can be computed in two different ways:

- as $\frac{\mathrm{WMC}_w(\Delta \wedge q)}{\mathrm{WMC}_w(\Delta)}$ in $B(\mathcal{L})$,
- and as $\mathrm{NWMC}_w([q])$ in $B(\Delta)$.

But how does the measure defined on $B(\mathcal{L})$ transfer to $B(\Delta)$?

4 Limitations of Factored Weighted Model Counting

F Give a concrete example of something impossible to represent using WMC.

F Can you say something here about factorized vs non-factorized weight function definitions? That is, factorized is when w maps literals to $R_{\geq 0}$, non-factorized is when w maps models to $R_{\geq 0}$ and

  – come up with nice example when non-factorized weights are intuitive;
  – clarify that the factorized definition have is w.r.t. models, in case some one gets confused. [It doesn't have to be, if the BA is not free—P.]

Lemma 6. *For any measure $m\colon \mathbf{B} \to \mathbb{R}_{\geq 0}$ and elements $a, b \in \mathbf{B}$,*

$$m(a \wedge b) = m(a)m(b) \tag{1}$$

if and only if

$$m(a \wedge b) \cdot m(\neg a \wedge \neg b) = m(a \wedge \neg b) \cdot m(\neg a \wedge b). \tag{2}$$

Proof. First, note that $a = (a \wedge b) \vee (a \wedge \neg b)$ and $(a \wedge b) \wedge (a \wedge \neg b) = 0$, so, by properties of a measure,

$$m(a) = m(a \wedge b) + m(a \wedge \neg b). \tag{3}$$

Applying Eq. (3) and the equivalent expression for $m(b)$ allows us to rewrite Eq. (1) as

$$m(a \wedge b) = [m(a \wedge b) + m(a \wedge \neg b)][m(a \wedge b) + m(\neg a \wedge b)]$$

which is equivalent to

$$m(a \wedge b)[1 - m(a \wedge b) - m(a \wedge \neg b) - m(\neg a \wedge b)] = m(a \wedge \neg b)m(\neg a \wedge b). \tag{4}$$

Since $a \wedge b$, $a \wedge \neg b$, $\neg a \wedge b$, $\neg a \wedge \neg b$ are pairwise disjoint and their supremum is 1,

$$m(a \wedge b) + m(a \wedge \neg b) + m(\neg a \wedge b) + m(\neg a \wedge \neg b) = 1,$$

and this allows us to rewrite Eq. (4) into Eq. (2). As all transformations are invertible, the two expressions are equivalent. □

This theorem needs a special case for zero weights.

Theorem 1. *Let $\mathbf{B}$ be a free BA over $\{l_i\}_{i=1}^n$ (for some $n \in \mathbb{N}$) with measure $m\colon \mathbf{B} \to \mathbb{R}_{\geq 0}$, and let $L = \{l_i\}_{i=1}^n \cup \{\neg l_i\}_{i=1}^n$. Then there exists a weight function $w\colon L \to \mathbb{R}_{\geq 0}$ such that $m = \mathrm{WMC}_w$ if and only if*

$$m(l \wedge l') = m(l)m(l') \tag{5}$$

for all distinct $l, l' \in L$ such that $l \neq \neg l'$.

Remark. Note that if $n = 1$, then Eq. (5) is vacuously satisfied and so any valid measure can be expressed as WMC.

Proof. ($\Leftarrow$) Let $w\colon L \to \mathbb{R}_{\geq 0}$ be defined by

$$w(l) = m(l) \tag{6}$$

for all $l \in L$. We are going to show that $\mathrm{WMC}_w = m$. First, note that $\mathrm{WMC}_w(0) = 0 = m(0)$ by the definitions of both WMC_w and m. Second, let

$$a = \bigwedge_{i=1}^n a_i \tag{7}$$

be an atom in $\mathbf{B}$ such that $a_i \in \{l_i, \neg l_i\}$ for all $i \in [n]$. Then

$$\mathrm{WMC}(a) = \prod_{i=1}^n w(a_i) = \prod_{i=1}^n m(a_i) = m\left(\bigwedge_{i=1}^n a_i\right) = m(a)$$

by Definition 6 and Eqs. (5) to (7). Finally, note that if WMC and m agree on all atoms, then they must also agree on all other non-zero elements of the Boolean algebra.

($\Rightarrow$) For the other direction, we are given a weight function $w\colon L \to \mathbb{R}_{\geq 0}$ that induces a measure $m = \mathrm{WMC}_w\colon \mathbf{B} \to \mathbb{R}_{\geq 0}$, and we want to show that Eq. (5) is satisfied. Let $k_i, k_j \in L$ be such that $k_i \in \{l_i, \neg l_i\}$, $k_j \in \{l_j, \neg l_j\}$, and $i \neq j$ for some $i, j \in [n]$. We then want to show that

$$m(k_i \wedge k_j) = m(k_i)m(k_j) \tag{8}$$

which is equivalent to

$$m(k_i \wedge k_j) \cdot m(\neg k_i \wedge \neg k_j) = m(k_i \wedge \neg k_j) \cdot m(\neg k_i \wedge k_j) \tag{9}$$

by Lemma 6. Then

$$\begin{aligned}
\mathrm{WMC}(k_i \wedge k_j) &= \sum_{\text{atoms } a \leq k_i \wedge k_j} \mathrm{WMC}(a) = \sum_{\text{atoms } a \leq k_i \wedge k_j} \prod_{m \in [n]} w(a_m) \\
&= \sum_{\text{atoms } a \leq k_i \wedge k_j} w(a_i)w(a_j) \prod_{m \in [n] \setminus \{i,j\}} w(a_m) = \sum_{\text{atoms } a \leq k_i \wedge k_j} w(k_i)w(k_j) \prod_{m \in [n] \setminus \{i,j\}} w(a_m) \\
&= w(k_i)w(k_j) \sum_{\text{atoms } a \leq k_i \wedge k_j} \prod_{m \in [n] \setminus \{i,j\}} w(a_m) = w(k_i)w(k_j)C,
\end{aligned}$$

where C denotes the part of $\mathrm{WMC}(k_i \wedge k_j)$ that will be the same for $\mathrm{WMC}(\neg k_i \wedge k_j)$, $\mathrm{WMC}(k_i \wedge \neg k_j)$, and $\mathrm{WMC}(\neg k_i \wedge \neg k_j)$ as well. But then Eq. (9) becomes

$$w(k_i)w(k_j)w(\neg k_i)w(\neg k_j)C^2 = w(k_i)w(\neg k_j)w(\neg k_i)w(k_j)C^2$$

which is trivially true. $\square$

Given this requirement for independence, a well-known way to represent probability distributions that do not consist entirely of independent variables is by adding more literals [12], i.e., extending the set L covered by the WMC weight function $w\colon L \to \mathbb{R}_{\geq 0}$. More precisely, we are given the left-hand column in

$$\begin{tikzcd}
\mathbb{R}_{\geq 0} & & \\
\mathbf{B} \arrow[u, "m"] \arrow[r, hook, "\iota"] & \mathbf{B}' \arrow[ul, "m'"'] & \\
L \arrow[u, hook] \arrow[r, hook] & L' \arrow[u, hook] \arrow[r, "w"] & \mathbb{R}_{\geq 0}
\end{tikzcd}$$

and construct the remaining part in such a way that the triangle commutes.

Must quotient the BA w.r.t. some rules.

5 Previous Work

5.1 Bayesian Network Encodings

- `cd05` relaxes the encoding so much that extra models become possible. They are supposed to be filtered out by the algorithm, but mine can't do that because it doesn't deal with models. Same for `cd06` because it's based on `cd05`.
- `sbk05` uses my trick with dividing probabilities. That could explain small inaccuracies in its answers.
- Encodings:
  - `d02` [15]

  - `sbk05` [45]
  - `cd05` [9]
  - `cd06` [10] (supposed to be the best)
  - `db20` (mine)

5.2 Algebraic Decision Diagrams and ADDMC

References

- ADDs [2]
- background reading
  - Compiling Bayesian Networks Using Variable Elimination (Chavira and Darwiche) [11]
  - On the Relationship between Sum-Product Networks and Bayesian Networks (Zhao et al.) [52]

6 Encoding Bayesian Networks Using Conditional Weights

- We assume that all variables in the Bayesian network have at least two values.
- Have an example of how the ADDs function in this situation. If not for the paper, then at least for slides. Use the framework to check its correctness.
- The function ϕ created by the algorithm can be seen as a measure on the BA 2^{2^U}.
- We extend the Gaifman graph to add edges when two variables occur in the same CPT (e.g., including the edge from A to B when the CPT is $\Pr(A \mid B)$).

Let V denote the set of random variables in a Bayesian network. For any random variable $X \in V$, let $\mathrm{pa}(X)$ denote the set of parents of X and $\mathrm{im}\, X$ denote the set of possible values.

Definition 7 (Indicator variables)**.** Let $X \in V$ be a random variable. If X is binary (i.e., $|\mathrm{im}\, X| = 2$), we can arbitrary identify one of the values as 1 and the other one as 0 (i.e, $\mathrm{im}\, X \cong \{0, 1\}$). Then X can be represented by a single *indicator variable* $\lambda_{X=1}$. For notational simplicity, for any set S, whenever we write $\lambda_{X=0} \in S$ or $S = \{\lambda_{X=0}, \dots\}$, we actually mean $\lambda_{X=1} \notin S$,

On the other hand, if X is not binary, we represent X with $|\mathrm{im}\, X|$ indicator variables, one for each value. We let

$$E(X) = \begin{cases} \{\lambda_{X=1}\} & \text{if } |\mathrm{im}\, X| = 2 \\ \{\lambda_{X=x} \mid x \in \mathrm{im}\, X\} & \text{otherwise.} \end{cases}$$

denote the set of indicator variables for X and

$$E^*(X) = E(X) \cup \bigcup_{Y \in \mathrm{pa}(X)} E(Y).$$

denote the set of indicator variables for X and its parents in the Bayesian network. Finally, let

$$U = \bigcup_{X \in V} E(X)$$

denote the set of all indicator variables for all random variables in the Bayesian network.

```
φ ← 1;
for X ∈ V do
    let pa(X) = {Y_1, ..., Y_n};
    CPT_X ← 0;
    if |im X| = 2 then
        for (y_1, ..., y_n) ∈ ∏_{i=1}^n im Y_i do
            p_1 ← Pr(X = 1 | Y_1 = y_1, ..., Y_n = y_n);
            p_0 ← Pr(X ≠ 1 | Y_1 = y_1, ..., Y_n = y_n);
            CPT_X ← CPT_X + p_1[λ_{X=1}] · ∏_{i=1}^n [λ_{Y_i=y_i}] + p_0 \overline{[λ_{X=1}]} · ∏_{i=1}^n [λ_{Y_i=y_i}];
    else
        let im X = {x_1, ..., x_m};
        for x ∈ im X and (y_1, ..., y_n) ∈ ∏_{i=1}^n im Y_i do
            p_x ← Pr(X = x | Y_1 = y_1, ..., Y_n = y_n);
            CPT_X ← CPT_X + p_x[λ_{X=x}] · ∏_{i=1}^n [λ_{Y_i=y_i}] + \overline{[λ_{X=1}]} · ∏_{i=1}^n [λ_{Y_i=y_i}];
        CPT_X ← CPT_X · (∑_{i=1}^m [λ_{X=x_i}]) · ∏_{i=1}^m ∏_{j=i+1}^m (\overline{[λ_{X=x_i}]} + \overline{[λ_{X=x_j}]});
    φ ← φ · CPT_X;
return φ;
```

Lemma 7. *Let $X \in V$ be a random variable with parents* $\mathrm{pa}(X) = \{Y_1, \ldots, Y_n\}$. *Then* $\mathrm{CPT}_X \colon 2^{E^*(X)} \to \mathbb{R}_{\geq 0}$ *is such that for any $x \in \operatorname{im} X$ and $(y_1, \ldots, y_n) \in \prod_{i=1}^n \operatorname{im} Y_i$,*

$$\mathrm{CPT}_X(\{\lambda_{X=x}\} \cup \{\lambda_{Y_i=y_i} \mid i = 1, \ldots, n\}) = \Pr(X = x \mid Y_1 = y_1, \ldots, Y_n = y_n).$$

Proof. Let $\tau = \{\lambda_{X=x}\} \cup \{\lambda_{Y_i=y_i} \mid i = 1, \ldots, n\}$. If X is binary, then CPT_X is a sum of $2\prod_{i=1}^n |\operatorname{im} Y_i|$ terms, one for each possible assignment of values to variables $X, Y_1, \ldots, Y_n$. Exactly one of these terms is nonzero when applied to τ, and it is equal to $\Pr(X = x \mid Y_1 = y_1, \ldots, Y_n = y_n)$ by definition.

If X is not binary, then

$$\left(\sum_{i=1}^m [\lambda_{X=x_i}]\right)(\tau) = 1,$$

and

$$\left(\prod_{i=1}^m \prod_{j=i+1}^m (\overline{[\lambda_{X=x_i}]} + \overline{[\lambda_{X=x_j}]})\right)(\tau) = 1,$$

so, by a similar argument as before,

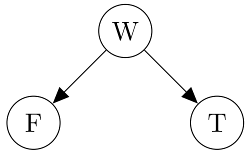
$$\mathrm{CPT}_X(\tau) = \Pr(X = x \mid Y_1 = y_1, \ldots, Y_n = y_n).$$

□

Proposition 1. *$\phi \colon 2^U \to \mathbb{R}_{\geq 0}$ represents the full probability distribution of the Bayesian network, i.e., if $V = \{X_1, \ldots, X_n\}$, then*

$$\phi(\tau) = \begin{cases} \Pr(X_1 = x_1, \ldots, X_n = x_n) & \text{if } \tau = \{\lambda_{X_i=x_i} \mid i = 1, \ldots, n\} \text{ for some } (x_1, \ldots, x_n) \in \prod_{i=1}^n \operatorname{im} X_i \\ 0 & \text{otherwise,} \end{cases}$$

for all $\tau \in 2^U$.



w	$\Pr(W = w)$
1	0.5
0	0.5

w	f	$\Pr(F = f \mid W = w)$
1	1	0.6
1	0	0.4
0	1	0.1
0	0	0.9

w	t	$\Pr(T = t \mid W = w)$
1	l	0.2
1	m	0.4
1	h	0.4
0	l	0.6
0	m	0.3
0	h	0.1

Figure 1: An example Bayesian network with its CPTs

Proof. If $\tau = \{\lambda_{X=v_X} \mid X \in V\}$ for some $(v_X)_{X \in V} \in \prod_{X \in V} \operatorname{im} X$, then

$$\phi(\tau) = \prod_{X \in V} \Pr\left(X = v_X \,\middle|\, \bigwedge_{Y \in \mathrm{pa}(X)} Y = v_Y\right) = \Pr\left(\bigwedge_{X \in V} X = v_X\right)$$

by Lemma 7 and the definition of a Bayesian network. Otherwise there must be some non-binary random variable $X \in V$ such that $|E(X) \cap \tau| \neq 1$. If $E(X) \cap \tau = \emptyset$, then

$$\left(\sum_{i=1}^{m} [\lambda_{X=x_i}]\right)(\tau) = 0,$$

and so $\mathrm{CPT}_X(\tau) = 0$, and $\phi(\tau) = 0$. If $|E(X) \cap \tau| > 1$, then we must have two different values $x_1, x_2 \in \operatorname{im} X$ such that $\{\lambda_{X=x_1}, \lambda_{X=x_2}\} \subseteq \tau$ which means that

$$(\overline{[\lambda_{X=x_1}]} + \overline{[\lambda_{X=x_2}]})(\tau) = 0,$$

and so, again, $\mathrm{CPT}_X(\tau) = 0$, and $\phi(\tau) = 0$. □

Theorem 2. *Let $\phi\colon 2^U \to \mathbb{R}_{\geq 0}$ be a function generated by the algorithm. Then*

$$(\exists_U (\phi \cdot [\lambda_{X=x}]))(\emptyset) = \Pr(X = x).$$

Proof. Let $V = \{X, Y_1, \ldots, Y_n\}$. Then

$$\begin{aligned}(\exists_U (\phi \cdot [\lambda_{X=x}]))(\emptyset) &= \sum_{\tau \in 2^U} (\phi \cdot [\lambda_{X=x}])(\tau) = \sum_{\lambda_{X=x} \in \tau \in 2^U} \phi(\tau) = \sum_{\lambda_{X=x} \in \tau \in 2^U} \left(\prod_{Y \in V} \mathrm{CPT}_Y\right)(\tau) \\ &= \sum_{(y_1, \ldots, y_n) \in \prod_{i=1}^{n} \operatorname{im} Y_i} \Pr(X = x, Y_1 = y_1, \ldots, Y_n = y_n) = \Pr(X = x)\end{aligned}$$

by the following arguments:

- the proof of Theorem 1 in the ADDMC paper [18];
- if $\lambda_{X=x} \notin \tau \in 2^U$, then $(\phi \cdot [\lambda_{X=x}])(\tau) = \phi(\tau) \cdot [\lambda_{X=x}](\tau \cap \{\lambda_{X=x}\}) = \phi(\tau) \cdot 0 = 0$;
- Proposition 1;
- marginalisation of a probability distribution.

□

Example 2. The Bayesian network in Fig. 1 has

$$
\begin{aligned}
V &= \{W, F, T\},\\
\mathrm{pa}(W) &= \emptyset,\\
\mathrm{pa}(F) &= \mathrm{pa}(T) = \{W\},\\
\operatorname{im} W &= \operatorname{im} F = \{0, 1\},\\
\operatorname{im} T &= \{l, m, h\},\\
E(W) &= \{\lambda_{W=1}\},\\
E(F) &= \{\lambda_{F=1}\},\\
E(T) &= \{\lambda_{T=l}, \lambda_{T=m}, \lambda_{T=h}\},\\
E^*(W) &= \{\lambda_{W=1}\},\\
E^*(F) &= \{\lambda_{F=1}, \lambda_{W=1}\},\\
E^*(T) &= \{\lambda_{T=l}, \lambda_{T=m}, \lambda_{T=h}, \lambda_{W=1}\},\\
\mathrm{CPT_W} &= 0.5[\lambda_{W=1}] + 0.5\overline{[\lambda_{W=1}]} = 0.5 \cdot 1,\\
\mathrm{CPT_F} &= 0.6[\lambda_{F=1}] \cdot [\lambda_{W=1}] + 0.4[\lambda_{F=0}] \cdot [\lambda_{W=1}] + 0.1[\lambda_{F=1}] \cdot [\lambda_{W=0}] + 0.9[\lambda_{F=0}] \cdot [\lambda_{W=0}]\\
&= 0.6[\lambda_{F=1}] \cdot [\lambda_{W=1}] + 0.4\overline{[\lambda_{F=1}]} \cdot [\lambda_{W=1}] + 0.1[\lambda_{F=1}] \cdot \overline{[\lambda_{W=1}]} + 0.9\overline{[\lambda_{F=1}]} \cdot \overline{[\lambda_{W=1}]},\\
\mathrm{CPT_T} &= ([\lambda_{T=l}] + [\lambda_{T=m}] + [\lambda_{T=h}]) \cdot (\overline{[\lambda_{T=l}]} + \overline{[\lambda_{T=m}]}) \cdot (\overline{[\lambda_{T=l}]} + \overline{[\lambda_{T=h}]}) \cdot (\overline{[\lambda_{T=m}]} + \overline{[\lambda_{T=h}]}) \cdot (\dots),
\end{aligned}
$$

and can be encoded in a DIMACS-like CNF format as

$$
\begin{array}{lrrrl}
\lambda_{T=l} & \lambda_{T=m} & \lambda_{T=h} & & 0\\
 & -\lambda_{T=l} & -\lambda_{T=m} & & 0\\
 & -\lambda_{T=l} & -\lambda_{T=h} & & 0\\
 & -\lambda_{T=m} & -\lambda_{T=h} & & 0\\
w & \lambda_{W=1} & & 0.5 & 0.5\\
w & \lambda_{F=1} & \lambda_{W=1} & 0.6 & 0.4\\
w & \lambda_{F=1} & -\lambda_{W=1} & 0.1 & 0.9\\
w & \lambda_{T=l} & \lambda_{W=1} & 0.2 & 1\\
w & \lambda_{T=m} & \lambda_{W=1} & 0.4 & 1\\
w & \lambda_{T=h} & \lambda_{W=1} & 0.4 & 1\\
w & \lambda_{T=l} & \lambda_{W=0} & 0.6 & 1\\
w & \lambda_{T=m} & \lambda_{W=0} & 0.3 & 1\\
w & \lambda_{T=h} & \lambda_{W=0} & 0.1 & 1
\end{array}
$$

with each λ replaced with a unique positive integer.

The last two numbers are the positive and the negative probabilities, respectively—sometimes they add to one, and sometimes the negative probability is one, regardless of the value of the first probability.

7 Experimental Comparison

- We don't compare 'compile times' because our encoding time is linear, so we would easily beat everyone else.
- When the Bayesian network has an evidence file, we compute the probability of evidence. Otherwise, let X denote the last-mentioned node in the Bayesian network. If `true` is a valid value of X, we compute the marginal probability of $X =$ `true`. Otherwise, we pick the first value of X and calculate its marginal probability. This applies to the Grid data set (as intended) and also to two instances of Plan Reconstruction and roughly half of the instances from 2004-PGM that have empty evidence files.

- After the experiments are finished, note the processor, memory per thread, and add the following acknowledgment.
- All other encodings are implemented in Ace 3.0[4] and should be compiled with `-encodeOnly` (i.e., don't compile the CNF into an AC) and `-noEclause` (i.e., only use standard syntax) flags.
- Datasets
  - binary Bayesian networks from Sang et al.[5] [45]
    * Grid (networks) (ratio 75 means that 75% of the nodes are deterministic),
    * Plan recognition (problems),
    * Deterministic quick medical reference (what do the numbers mean? the README doesn't say).
  - Bayesian networks available with Ace
    * `2004-pgm` [13] (binary)
    * `2005-ijcai` [9]. The Genie/Smile files have their own citation data that I should probably extract. This is the only dataset that has some non-binary networks.
    * `2006-ijar` [13] (binary)

8 Explaining The Performance Benefits

- `d02` has
$$\sum_{X \in V} |\operatorname{im} X| + |\operatorname{im} X| \prod_{Y \in \operatorname{pa}(X)} |\operatorname{im} Y|$$
variables and
$$\sum_{X \in V} 1 + \binom{|\operatorname{im} X|}{2} + |\operatorname{im} X|(2 + |\operatorname{pa}(X)|) \prod_{Y \in \operatorname{pa}(X)} |\operatorname{im} Y|$$
clauses (along with one ADD per variable to encode the weights).
- `sbk05` is a bit harder to evaluate due to a handful of small optimisations in the encoding. Could find an upper bound anyway.
- `db20` (my encoding) has
$$\sum_{X \in V} |\operatorname{im} X|$$
variables (less for binary) and
$$\sum_{X \in V} |\operatorname{im} X| + 1 + \binom{|\operatorname{im} X|}{2}$$
ADDs.
- Let:
  - $N = |V|$ (i.e., the number of nodes in the Bayesian network),
  - $D = \max_{X \in V} |\operatorname{pa}(X)|$ (i.e., the maximum in-degree or the number of parents),
  - $V = \max_{X \in V} |\operatorname{im} X|$ (i.e., the maximum number of values per variables).

[4] `http://reasoning.cs.ucla.edu/ace/`
[5] `https://www.cs.rochester.edu/u/kautz/Cachet/`

- Then my encoding has $\mathcal{O}(NV)$ variables and $\mathcal{O}(NV^2)$ ADDs while `d02` has $\mathcal{O}(NV^{D+1})$ variables and $\mathcal{O}(NDV^{D+1})$ ADDs.

Calculate numVariables/numClauses (or the other way around) for each instance and plot this ratio vs runtime (for each encoding, or at least mine and D02)

9 Conclusion and Future Work

- Bayesian networks and ADDMC are only particular examples. This should also work with Cachet.
- Extra benefit: one does not need to come up with a way to turn some probability distribution to into a fully independent one.
- Important future work: replacing ADDs with AADDs[6] [46] is likely to bring performance benefits. Other extensions:
  - FOADDs can represent first order statements;
  - XADDs can replace WMI for continuous variables;
  - ADDs with intervals can do approximations.
- Filtering out ADDs that have nothing to do with the answer helps tremendously, but I'm purposefully not doing that. Perhaps a heuristic could do the same thing?
- Encodings for everything else
  - probabilistic programs [26]
  - ProbLog [19]
    * For the ProbLog to WMC conversion, check out this guy: `https://users.ics.aalto.fi/ttj/`.
    * proof-based [38]
    * rule-based [28]
    * For ground ProbLog, we can encode a program
      ```
      p :: a :- b
      q :: a :- c
      ```
      into $P(a \mid b) = p$, $P(a \mid c) = q$ instead of having clauses $b \Rightarrow a$, $c \Rightarrow a$. Some logical structure is likely to remain.
- Bayesian networks are often solved in a compile once, query many times fashion. This can be achieved using ADDMC by selecting a subset S of variable we may want to query over and running ADDMC while excluding S from variable elimination/projection/$\exists$.
- More references
  - Measures on/in Boolean algebras: Horn and Tarski [27], Jech [30]
  - On Boolean algebras and their role in analysis [50]
  - Infinite domains
    * Markov Logic in Infinite Domains (Singla and Domingos) [48]
    * Objective Bayesian probabilistic logic (Williamson) [49]
    * Unifying Logic and Probability (Russell) [43]

[6] `https://github.com/ssanner/dd-inference`

- Logical induction [22]
- Quantum probabilistic logic programming [3]
- WMC
  * algebraic model counting [32]
  * Explanation-Based Approximate Weighted Model Counting for Probabilistic Logics [42]
  * OUWMC [4]
  * Formula-Based Probabilistic Inference [24]
  * Parallel Probabilistic Inference by WMC [14]
  * Semiring Programming [6]
  * theoretical extension: WMC beyond two-variable logic [35]
  * from weighted to unweighted model counting [8]
  * theory behind WMC algorithms: solving #SAT and Bayesian inference with backtracking search [1]

Acknowledgements. This work has made use of the resources provided by the Edinburgh Compute and Data Facility (ECDF) (`http://www.ecdf.ed.ac.uk/`).

References

[1] Fahiem Bacchus, Shannon Dalmao, and Toniann Pitassi. Solving #sat and bayesian inference with backtracking search. *J. Artif. Intell. Res.*, 34:391–442, 2009.

[2] R. Iris Bahar, Erica A. Frohm, Charles M. Gaona, Gary D. Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. Algebraic decision diagrams and their applications. *Formal Methods Syst. Des.*, 10(2/3):171–206, 1997.

[3] Radhakrishnan Balu. Quantum probabilistic logic programming. In *Quantum Information and Computation XIII*, volume 9500, page 950011. International Society for Optics and Photonics, 2015.

[4] Vaishak Belle. Open-universe weighted model counting. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 3701–3708. AAAI Press, 2017.

[5] Vaishak Belle. Weighted model counting with function symbols. In Gal Elidan, Kristian Kersting, and Alexander T. Ihler, editors, *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press, 2017.

[6] Vaishak Belle and Luc De Raedt. Semiring programming: A framework for search, inference and learning. *CoRR*, abs/1609.06954, 2016.

[7] Elena Castiñeira, Susana Cubillo, and Enric Trillas. On possibility and probability measures in finite Boolean algebras. *Soft Comput.*, 7(2):89–96, 2002.

[8] Supratik Chakraborty, Dror Fried, Kuldeep S. Meel, and Moshe Y. Vardi. From weighted to unweighted model counting. In Yang and Wooldridge [51], pages 689–695.

[9] Mark Chavira and Adnan Darwiche. Compiling Bayesian networks with local structure. In Kaelbling and Saffiotti [31], pages 1306–1312.

[10] Mark Chavira and Adnan Darwiche. Encoding CNFs to empower component analysis. In Armin Biere and Carla P. Gomes, editors, *Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings*, volume 4121 of *Lecture Notes in Computer Science*, pages 61–74. Springer, 2006.

[11] Mark Chavira and Adnan Darwiche. Compiling bayesian networks using variable elimination. In Manuela M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 2443–2449, 2007.

[12] Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. *Artif. Intell.*, 172(6-7):772–799, 2008.

[13] Mark Chavira, Adnan Darwiche, and Manfred Jaeger. Compiling relational Bayesian networks for exact inference. *Int. J. Approx. Reason.*, 42(1-2):4–20, 2006.

[14] Giso H. Dal, Alfons W. Laarman, and Peter J. F. Lucas. Parallel probabilistic inference by weighted model counting. In Milan Studený and Václav Kratochvíl, editors, *International Conference on Probabilistic Graphical Models, PGM 2018, 11-14 September 2018, Prague, Czech Republic*, volume 72 of *Proceedings of Machine Learning Research*, pages 97–108. PMLR, 2018.

[15] Adnan Darwiche. A logical approach to factoring belief networks. In Dieter Fensel, Fausto Giunchiglia, Deborah L. McGuinness, and Mary-Anne Williams, editors, *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France, April 22-25, 2002*, pages 409–420. Morgan Kaufmann, 2002.

[16] Adnan Darwiche. New advances in compiling CNF into decomposable negation normal form. In de Mántaras and Saitta [17], pages 328–332.

[17] Ramón López de Mántaras and Lorenza Saitta, editors. *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004*. IOS Press, 2004.

[18] Jeffrey M. Dudek, Vu Phan, and Moshe Y. Vardi. ADDMC: weighted model counting with algebraic decision diagrams. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 1468–1476. AAAI Press, 2020.

[19] Daan Fierens, Guy Van den Broeck, Ingo Thon, Bernd Gutmann, and Luc De Raedt. Inference in probabilistic logic programs using weighted CNF's. In Fábio Gagliardi Cozman and Avi Pfeffer, editors, *UAI 2011, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, Barcelona, Spain, July 14-17, 2011*, pages 211–220. AUAI Press, 2011.

[20] Haim Gaifman. Concerning measures on Boolean algebras. *Pacific Journal of Mathematics*, 14(1):61–73, 1964.

[21] M. Ganesh. *Introduction to fuzzy sets and fuzzy logic.* PHI Learning Pvt. Ltd., 2006.

[22] Scott Garrabrant, Tsvi Benson-Tilsen, Andrew Critch, Nate Soares, and Jessica Taylor. Logical induction. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:154, 2016.

[23] Steven Givant and Paul R. Halmos. *Introduction to Boolean algebras.* Springer Science & Business Media, 2008.

[24] Vibhav Gogate and Pedro M. Domingos. Formula-based probabilistic inference. In Peter Grünwald and Peter Spirtes, editors, *UAI 2010, Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, July 8-11, 2010*, pages 210–219. AUAI Press, 2010.

[25] Theodore Hailperin. Probability logic. *Notre Dame Journal of Formal Logic*, 25(3):198–212, 1984.

[26] Steven Holtzen, Guy Van den Broeck, and Todd D. Millstein. Dice: Compiling discrete probabilistic programs for scalable inference. *CoRR*, abs/2005.09089, 2020.

[27] Alfred Horn and Alfred Tarski. Measures in Boolean algebras. *Transactions of the American Mathematical Society*, 64(3):467–497, 1948.

[28] Tomi Janhunen. Representing normal programs with clauses. In de Mántaras and Saitta [17], pages 358–362.

[29] Thomas Jech. *Set theory, Second Edition.* Perspectives in Mathematical Logic. Springer, 1997.

[30] Thomas Jech. Measures on Boolean algebras. *arXiv preprint arXiv:1705.01006*, 2017.

[31] Leslie Pack Kaelbling and Alessandro Saffiotti, editors. *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*. Professional Book Center, 2005.

[32] Angelika Kimmig, Guy Van den Broeck, and Luc De Raedt. Algebraic model counting. *J. Appl. Log.*, 22:46–62, 2017.

[33] Sabine Koppelberg, Robert Bonnet, and James Donald Monk. *Handbook of Boolean algebras*, volume 384. North-Holland Amsterdam, 1989.

[34] Peter H. Krauss. Representation of conditional probability measures on Boolean algebras. *Acta Mathematica Hungarica*, 19(3-4):229–241, 1968.

[35] Antti Kuusisto and Carsten Lutz. Weighted model counting beyond two-variable logic. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 619–628. ACM, 2018.

[36] Jean-Marie Lagniez and Pierre Marquis. An improved decision-DNNF compiler. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 667–673. ijcai.org, 2017.

[37] Ken Levasseur and Al Doerr. *Applied Discrete Structures*. Lulu.com, 2012.

[38] Theofrastos Mantadelis and Gerda Janssens. Dedicated tabling for a probabilistic setting. In Manuel V. Hermenegildo and Torsten Schaub, editors, *Technical Communications of the 26th International Conference on Logic Programming, ICLP 2010, July 16-19, 2010, Edinburgh, Scotland, UK*, volume 7 of *LIPIcs*, pages 124–133. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010.

[39] Nils J. Nilsson. Probabilistic logic. *Artif. Intell.*, 28(1):71–87, 1986.

[40] Nils J. Nilsson. Probabilistic logic revisited. *Artif. Intell.*, 59(1-2):39–42, 1993.

[41] Umut Oztok and Adnan Darwiche. A top-down compiler for sentential decision diagrams. In Yang and Wooldridge [51], pages 3141–3148.

[42] Joris Renkens, Angelika Kimmig, Guy Van den Broeck, and Luc De Raedt. Explanation-based approximate weighted model counting for probabilistic logics. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 2490–2496. AAAI Press, 2014.

[43] Stuart J. Russell. Unifying logic and probability. *Commun. ACM*, 58(7):88–97, 2015.

[44] Tian Sang, Fahiem Bacchus, Paul Beame, Henry A. Kautz, and Toniann Pitassi. Combining component caching and clause learning for effective model counting. In *SAT 2004 - The Seventh International Conference on Theory and Applications of Satisfiability Testing, 10-13 May 2004, Vancouver, BC, Canada, Online Proceedings*, 2004.

[45] Tian Sang, Paul Beame, and Henry A. Kautz. Performing Bayesian inference by weighted model counting. In Manuela M. Veloso and Subbarao Kambhampati, editors, *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, pages 475–482. AAAI Press / The MIT Press, 2005.

[46] Scott Sanner and David A. McAllester. Affine algebraic decision diagrams (aadds) and their application to structured probabilistic inference. In Kaelbling and Saffiotti [31], pages 1384–1390.

[47] Roman Sikorski. *Boolean algebras*. Springer, third edition, 1969.

[48] Parag Singla and Pedro M. Domingos. Markov logic in infinite domains. In Ronald Parr and Linda C. van der Gaag, editors, *UAI 2007, Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence, Vancouver, BC, Canada, July 19-22, 2007*, pages 368–375. AUAI Press, 2007.

[49] Jon Williamson. Objective bayesian probabilistic logic. *J. Algorithms*, 63(4):167–183, 2008.

[50] Katarzyna Winkowska-Nowak. *On Boolean algebras and their role in analysis*. PhD thesis, Florida Atlantic University, 1996.

[51] Qiang Yang and Michael J. Wooldridge, editors. *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. AAAI Press, 2015.

[52] Han Zhao, Mazen Melibari, and Pascal Poupart. On the relationship between sum-product networks and bayesian networks. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 116–124. JMLR.org, 2015.